MASTER

Softwareentwicklung

Endbericht

Architekturerstellung auf Basis von Anforderungen

> So vielfältig kann Technik sein.

FACHHOCHSCHULE
TECHNIKUM WIEN

FACHHOCHSCHULE TECHNIKUM WIEN

Ziel

Das Ziel der Arbeit dreht sich um die Beantwortung folgender folgender Fragen:

- Wie kommt man von Anforderungen auf eine gute Architektur?
- Gibt es eine Art Kochrezept für die Architekturerstellung?

Methoden

Um diese Fragen zu beantworten wurde ein Projekt durchgeführt. Bei diesem Projekt handelt es sich um das Erstellen einer Architektur des IT Systems einer Zertifizierungsstelle.

Dazu wurden die Anforderungen in einem erweitertem Anforderungstemplate aufgenommen welches aus einem Usecase Diagramm abgeleitet wurde. Da die Grundannahme darin bestand, dass eine gute Architektur sich nur aus guten nicht funktionalen Anforderungen ableiten lässt wurden in diesem Template zusätzliche nicht funktionale Anforderungen aufgenommen. Außerdem wurde versucht, schon früh Architekturreview relevante Parameter zu dokumentieren. Folgende Punkte wurden hinzugefügt:

- Earned Value pro Monat: Wieviel Umsatz generiert der Usecase
- System Access: Welche Zugrifssrichtungen besitzt der Usecase (z.B. Ingoing Private)
- Possible Extensions: Mögliche Änderungen und zusätzliche Funktionalität
- Expected Usage: Wieviele Nutzer/Zugriffe pro Zeiteinheit zu erwarten sind
- Growth Scenarios: Wieviele Nutzer/Zugriffe pro Zeiteinheit möglich sein könnten
- Non Functional Requirements: Auflistung und Gewichtung von nicht funktionalen Attributen auf einer Skala von 1 (wenig wichtig) bis 3 (sehr wichtig)

Danach wurden mehrere Anläufe gestartet um von den Anforderungen auf eine gute Architektur zu kommen.

FACHHOCHSCHULE TECHNIKUM WIEN

Erster Anlauf der Prozesserstellung

Der erste Anlauf bestand darin, pro Usecase ein System zu entwerfen, welches den nicht funktionalen Attributen entsprach. Als Beispiel hierfür ist der Anmelde Prozess, welcher besonders gut skalierbar und eine hohe Verfügbarkeit haben sollte. Es wurde ein klassisches Websystem geplant, komplett mit Webserver, Application Server, Database Server und Load Balancer.

Diese Herangehensweise sollte für alle anderen Usecases durchgeführt werden. Am Ende sollten alle Architekturen mit einander vereint werden und durch eine Prüfung der nicht funktionalen Parameter sollte entschieden werden, welche Komponenten zusammengelegt werden sollten.

Schnell wurde klar, dass diese Herangehensweise einen enormen Arbeits- und Modellierungsaufwand verursachte und wegen der enormen Vielfalt an Konfigurationen nicht eindeutig war.

Zweiter Anlauf der Prozesserstellung

Der zweite Anlauf war immer noch sehr auf nicht funktionale Anforderungen fokussiert und versuchte, die Probleme des ersten Prozesses, den enormen Aufwand, zu beseitigen.

Dazu wurde von einer groben Skizze der Architektur ausgegangen, welche auf Erfahrungswerten basiert erstellt wurde. Danach wurde versucht, durch die Prüfung der nicht funktionalen Anforderungen die Architektur zu verfeinern, zusätzliche Systeme zu erstellen oder bestehende zusammen zu legen.

Der große Schwachpunkt dieses Architekturprozesses lag in ursprünglichen Motivation, nämlich den Aufwand der Erstellung zu verringern: Durch die grobe Schätzung konnten keine Regeln abgeleitet werden; hätte ein andere die selbe Architektur nach diesem Prozess erstellt, wäre er vielleicht auf ein völlig anderes Ergebnis gekommen.

Dritter Anlauf der Prozesserstellung

Da das Usecase/Komponenten Konzept in den vorherigen Anläufen sich als zu ungenau oder aufwendig erwiesen hat, wurde eine andere Herangehensweise gewählt: Der Fokus auf funktionale Attribute, nämlich die Daten.

Die Daten wurden nach Vertrautheitslevel aufgespalten und in drei verschiedene Systeme geteilt, welche jeweils durch Gateways/Firewalls getrennt waren. Das Ergebnis war ein System mit fünf unterschiedlichen Komponenten, welches nicht mehr weiter definiert werden konnte.

Der Vierte und letzte Anlauf wurde darauf in folgender Weise realisiert.

Ergebnisse

Zuerst werden die Daten wie im dritten Anlauf in unterschiedliche Vertrautheitslevel eingeteilt und mit Gateways/Firewalls getrennt. Anschließend wurde das Selbe für die Akteure des Systems durchgeführt: jeder Akteur wurde in ein Vertrautheitslevel eingestuft.

Danach wurde anhand der Usecases und Aktivitätsdiagramme analysiert, welcher Akteur auf welche Daten Zugriff hatte. Akteure mit einem niedrigerem Vertrautheitslevel als die Daten selbst erhielten ein eigenes System, durch welches die Daten transportiert wurden. Gab es mehrere Akteure, die höherwertige Daten übertrugen oder auf höherwertige Daten Zugriff hatten wurden diese Systeme getrennt.

Eine zusätzliche Regel für die Gateways/Firewalls wurde aufgestellt: Daten können nur in höhere Systeme gelangen, wenn sie von bekannten Systemen ausgingen. Das ermöglichte es, die Aufspaltung in eigene Systeme nicht durch alle Level zu duplizieren. Weiters wurde folgende Regel erstellt: Ein Request von einem System zu einem anderen System muss immer durch alle Zwischensysteme gehen. Dies ermöglichte es, geschlossene Systeme mit klaren Grenzen zu definieren.

Diese Regeln wurden durch folgendes Beispiel abgeleitet: Ein Prüfer hatte zum Beispiel die Aufgabe, hoch vertrauenswürdige Prüfungsdaten von Außen zu übermitteln, daher erhielt er ein eigenes System. Die gleiche Überlegung fand auch für den Scheme Owner statt, welcher die Fragen an die Zertifizierungsstelle übermitteln sollte. Beide hatten von Außen Zugriff auf eine höhere Systemebene. Ein zusätzliches Angriffsszenario wurde durchgespielt: Was würde passieren, wenn der Scheme Owner die Daten des Prüfers einsehen könnte und umgekehrt? Dieses Szenario wurde als geschäftsschädigend bewertet und somit war es klar, dass Systeme, welche als Einstiegspunkt für Akteure mit niedrigerem Vertrautheitslevel als die Daten getrennt werden müssen.

Das Ergebnis dieses Anlaufs deckte sich gut mit dem Anfangs geschätztem System, hatte eine klare, strukturierte Vorgehensweise und war genau genug, um eine Übersicht über das zu erstellende Systeme zu erlangen. Da noch keine konkreten Komponenten fest gelegt wurden, aber die Interfaces und Kommunikationswege schon ersichtlich waren, ergab sich ein weiterer Vorteil: Die Entscheidung der genauen Implementation der einzelnen Systeme war nicht notwendig, um die Architektur zu erstellen, je nach Anforderung könnten die Systeme zum Beispiel als eine Cloud Architektur umgesetzt werden (z.B. die Webseite für die Anmeldungen) oder einfach nur als einzelner, Rasperry Pie Gateway Server (z.B. für die Übermittlung der Prüfungsdaten).

Ein entscheidender Teil fehlte jedoch: die nicht funktionalen Anforderungen wurde nicht berücksichtigt um die Architektur zu erstellen, sondern bisher drehte sich alles um die funktionalen Anforderungen wie z.B. Security oder fachliche Dinge wie Daten. Daher wurden nun auf der bestehenden Architektur versucht, nicht funktionale Attribute zu überprüfen.

Schnell fiel auf, dass zum Zeitpunkt der Planung nicht alle nicht funktionalen Anforderungen relevant waren: z.B. war Usability praktisch nicht relevant für die Architekturerstellung: Außerdem fehlte die Implementierung, um nicht funktionale Messungen wie z.B. Geschwindigkeit durch zu führen.

Da diese Werte aber sehr wertvoll wären um spätere Fehlerkosten aufgrund von falschen Entscheidungen zu vermeiden, wurde statt der expliziten Messung der Werte eine Analyse anhand von Erfahrungswerten durchgeführt.

Für Performance kann z.B. die Anzahl der betroffenen Systeme durch das Zählen von Swimlanes in den Aktivitätsdiagrammen gemessen werden und anschließend mit einem Durchschnittswert multipliziert werden, welcher für einen durchschnittlichen Request über das Netzwerk erwartet würde. Dieser Wert kann anschließend mit erwünschten Antwortzeiten verglichen werden.

Für die Verfügbarkeit können z.B. die Ausfallkosten geschätzt werden und mit der Investition in ein zusätzliches System verglichen werden um zu sehen, welche Ausfallsicherheit sich für die Komponenten rechnet.

Schlussfolgerungen

Durch die Durchführung des Projektes und die Erstellung des Prozesses ergaben sich folgende Erkenntnisse:

- Nicht funktionale Anforderungen bestimmen die Qualität der Architektur maßgeblich, sind jedoch nicht zu jeder Phase des Prozesses messbar. Sie sind zusätzlich weniger eher ungeeignet, um einen Architekturprozess abzuleiten
- Es ist grundsätzlich möglich, einen allgemeingültigen Architekturprozess zu erstellen
- Datenstrukturen und Assoziationen sind für die Erstellung der Architektur relevanter als gedacht
- Viele Aspekte der Architektur können schon während der Planungsphase einbezogen werden, um Fehlerkosten zu verringern oder zu vermeiden
- Architekturreviewdaten können zum Großteil schon beim Ermitteln der Anforderungen aufgezeichnet werden
- Durch den Fokus auf funktionale Anforderungen können wichtige Entscheidungen aufgeschoben werden

- Viele Paramter können über Diagramme abgeleitet werden, z.B. sind die Swimlanes der Aktivitätsdiagramme sehr vielseitig interpretier- und verwendbar
- Die Bewertung der nicht funktionale Anforderungen erfordert konkrete Parameter zur Überprüfung und ist stark abhängig von Erfahrungswerten